Appendix A

7. OM Implementation Details

7.1. OMCD

7.1.1. Data Structures

```
typedef
struct vpn_descriptor_s
{
   TAG_DECL;                                  /* run time type identification */
   dlcl_list_t        list;                   /* List of all VPNs */
   dlcl_list_t        vr_list;                /* list of all VR for this VPN */
   lck_mutex_t        vr_lock;
   uint32_t        id;                        /* Globally unique VPN id */
   int                numb_of_vrs;               /* Number of VR's in this VPN */
   char            name[NAMELEN+1];           /* VPN name */
   int                admin_status;              /* Administrative status of the
VPN */
#define VPN_ADM_CREATE        0x0001             /* event */
#define VPN_ADM_GO            0x0002             /* event */
#define VPN_ADM_SUSPEND       0x0004             /* event */
#define VPN_ADM_DESTROY       0x0008             /* event */
#define VPN_ADM_NOT_EXIST     0x0100             /* state */
#define VPN_ADM_INACTIVE      0x0200             /* state */
#define VPN_ADM_ACTIVE        0x0400             /* state */
   int                oper_status;            /* Operational status of the VPN */
#define VPN_OPER_DOWN            0x01
#define VPN_OPER_INIT            0x02
#define VPN_OPER_UP              0x04
#define VPN_OPER_FAULT_RECOVERY     0x08
   int                marked;    /* Used to delete vpn in CBR data base */
   long            timestamp;    /*  Time when data was send to CBR data
base */
}  vpn_descriptor_t;

typedef
struct vr_descriptor_s
{
   TAG_DECL;                                  /* run time type identification */
   dlcl_list_t        list;                   /* List of all VRs in a VPN */
   ipaddr_t        id;                        /* Identifies VR */
   vpn_descriptor_t  *vpn;                    /*  Identifies  VPN  to  which  VR
belongs */
   object_group_id_t group;    /* Identifies group to which VR mapped */
   int             admin_status;    /* Administrative status of the VR */
#define VR_ADM_CREATE         0x0001             /* event */
#define VR_ADM_GO          0x0002             /* event */
#define VR_ADM_SUSPEND        0x0004             /* event */
```

```
#define VR_ADM_DESTROY      0x0008          /* event */
#define VR_ADM_NOT_EXIST    0x0100          /* state */
#define VR_ADM_INACTIVE     0x0200          /* state */
#define VR_ADM_ACTIVE       0x0400          /* state */
   int            oper_status;   /* Operational status of the VR */
#define VR_OPER_DOWN        0x01
#define VR_OPER_INIT        0x02
#define VR_OPER_UP             0x04
#define VR_OPER_FAULT_RECOVERY   0x08
   int            marked;  /* Used to delete vpn in CBR data base */
   long           timestamp;       /*  Time when data was send to CBR
data base */
}  vr_descriptor_t;
```

7.1.2. API

7.1.2.1. Create / Destroy API

FUNCTION	EFFECT	ARGUMENTS
`om_create_vpn`	Create VPN with specified VPN ID. Returns SUCCESS or FAILURE.	VPN ID
`om_create_vr`	Create VR for existing VPN with specified VR ID. Returns SUCCESS or FAILURE.	VPN ID, VR ID
`om_destroy_vpn`	Destroys VPN and all VRs configured in the VPN. Returns SUCCESS or FAILURE.	VPN ID
`om_destroy_vr`	Destroys VR. Returns SUCCESS or FAILURE.	VPN ID, VR ID

7.1.2.2. IOCTL API

FUNCTION	EFFECT	ARGUMENTS
`om_ioctl`	Sends IOCTL with data to object of the specified class, which belongs to the VR, identified by < VPN ID, VR ID>. Returns SUCCESS or FAILURE in case of system failure or any other status supplied by the object-specific IOCTL handler.	VPN ID, VR ID, class id, IOCTL code, data pointer, length
`om_ioctl_by_obj_id`	Sends IOCTL with data to object identified by supplied object id. Returns SUCCESS or FAILURE in case of system failure or any other status supplied by the object-specific IOCTL handler.	Object ID IOCTL code, data pointer, length

7.1.2.3. Check existence

FUNCTION	EFFECT	ARGUMENTS
`om_is_vpn_exist`	Checks the existence of the VR, which is specified by the <VPN ID, VR ID>. Return TRUE if such a VPN exist and FALSE otherwise	VPN ID
`om_is_vr_exist`	Checks the existence of the VPN, which is specified by the VPN ID. Return TRUE if such a VPN exist and FALSE otherwise	VPN ID, VR ID

7.1.2.4. Attribute Access

FUNCTION	EFFECT	ARGUMENTS
`om_get_first_vpn`	Looks up first VPN ih the global list of VPNs in the single IPSX system. Returns ID of the first VPN if at least one VPN exists, and 0 otherwise	
`om_get_next_vpn`	Searches the global list of the VPNs on the single Orion for the next VPN after the one, specified by VPN ID. If VPN ID is 0, then it returns VPN ID of the first VPN. If the VPN specified by VPN ID does not exist, then function returns next larger VPN ID than specified one. Returns 0 if no VPN was created, or there is no VPN with VPN ID and there is no next VPN.	VPN ID
`om_get_first_vr`	Looks up first VR in the list of the VRs, which belong to the VPN with VPN ID in the single Orion. Returns pointer to the VR ID at least one VR exists, and NULL otherwise. Returns NULL if there is no VPN with supplied VPN ID.	VPN ID
`om_get_next_vr`	Looks up the next VR after one, specified by the VR ID in the list of the VRs, which belong to the VPN with VPN ID in the single Orion. If VPN ID is not valid, then function returns NULL. If VR with specified VR ID does not exist, then function returns a pointer to the VR with next valid VR ID. Returns NULL if no VR was created in VPN with	VPN ID, VR ID

	VPN ID, or there is no next VR after VR with specified VR ID.	
om_get_vpn_name	Gets a name of a VPN, specified by the VPN ID. SUCCESS is returned if the VPN with VPN ID was found and FAILURE is returned otherwise.	VPN ID, name pointer, name size
om_set_vpn_name	Sets a name of a VPN, specified by the VPN ID. SUCCESS is returned if the VPN with VPN ID was found and FAILURE is returned otherwise.	VPN ID, name pointer, name size
om_get_vr_name	Gets a name of a VR, specified by the VPN ID and VR ID. SUCCESS is returned if the VR was found and FAILURE is returned otherwise.	VPN ID, VR ID, name pointer, name size
om_set_vr_name	Sets a name of a VR, specified by the <VPN ID, VR ID>. SUCCESS is returned if the VR was found and FAILURE is returned otherwise.	VPN ID, VR ID, name pointer, name size
om_get_vr_type	Gets a type of a VR, specified by the VPN ID and VR ID. SUCCESS is returned if the VR was found and FAILURE is returned otherwise.	VPN ID, VR ID, VR type
om_set_vr_type	Sets a type of a VR, specified by the <VPN ID, VR ID>. SUCCESS is returned if the VR was found and FAILURE is returned otherwise.	VPN ID, VR ID, VR type

7.2. OMORIG

7.2.1. Data Structures

```
typedef
struct oid_link_s
{
   TAG_DECL;
   dlcl_list_t      grp_lst;       /* List of all object IDs in the
group */
   dlcl_list_t      omori_lst;     /* List of all object IDs in the
group */
   dlcl_list_t      global_lst;    /* Sorted Global List of all object
IDs in the system */
   dlcl_list_t      list;          /* Global List of all object IDs in
the system */
   object_id_t      id;
```

```
   int              marked;         /* Used to delete oid in CBR data
base */
   long          timestamp;         /* Time when data was send to CBR
data base */
}  oid_link_t;
```

7.2.2. Message Tags

```
/* OMORI and OMORIG reply tag */
#define  OM_REPLY_TAG                 0x4000
The following tags are sent by OMORIG to OMORI
#define  OMORI_DESTROY_GRP_OBJS_TAG  0x4003
#define  OMORI_MV_OBJ_TO_GRP_TAG     0x4005
#define  OMORI_CREATE_OBJ_TAG        0x4009
#define  OMORI_DESTROY_OBJ_TAG       0x4011
#define  OMORI_ACTIVATE_OBJ_TAG      0X4013
#define  OMORI_STOP_OBJ_TAG          0x4015
#define  OMORI_FORWARD_OBJ_ID_TAG    0x4019
#define  OMORI_IOCTL_TAG             0x4021
#define  OMORI_UPDATE_VR_ATTR_TAG    0x4023
#define  OMORI_UPDATE_GRP_TAG        0x4025
#define  OMORI_UPDATE_OBJ_TAG        0x4027
```

7.2.3. API

7.2.3.1. Group Membership Change

FUNCTION	EFFECT	ARGUMENTS
`omorig_add_obj_id`	Adds object ID to the OM Global database. Returns SUCCESS or FAILURE	Object ID
`omorig_remove_obj_id`	Removes object ID from the OM Global database. Returns SUCCESS or FAILURE	Object ID

7.2.3.2. Attribute Access

FUNCTION	EFFECT	ARGUMENTS
`omorig_get_first_group`	Returns pointer to the first group in the DB	Pointer to the group hash table
`omorig_get_next_group`	Returns pointer to the next after the specified group in the DB	Pointer to the group hash table and the pointer to the group
`omorig_lookup_group_by_id`	Looks up group by the specified group ID. Returns pointer to the group descriptor if found and	GROUP ID

	NULL otherwise.	
`omorig_lookup_oid`	Looks up OID link in the DB by specified ID. Returns pointer to the OID link if found and NULL otherwise.	Object ID

7.3. OMORI

7.3.1. Data Structures

```
typedef
struct obj_descriptor_s
{
   TAG_DECL;                            /* run time type identification */
   dlcl_list_t    global_list;          /* Global list of all objects */
   dlcl_list_t    list;          /* List of all objects in the group */
   object_id_t    id;    /* Unique object id in the group context */
   int            class_id;          /* Identifies class of the object */
   uint32_t       vpn_id;    /* Identifies VPN to which object belongs */
   ipaddr_t       vr_id;          /* Identifies VR */
   global_address_t  obj_instance;      /*   Identifies   the   object   in
different address space */
   uint32_t       timestamp;      /* Time when object was registered */
} obj_descriptor_t;
```

7.3.2. Message Tags

The following tags are sent by OMORI to OMORIG

```
#define  OMORIG_ADD_OBJ_ID_TAG        0x5005
#define  OMORIG_RM_OBJ_ID_TAG         0x5006
#define  OMORIG_MV_OBJ_ID_TAG         0x5007
#define  OMORIG_FORWARD_OBJ_ID_TAG    0x5009
#define  OMORIG_REGISTER_FUNS_TAG     0x5011
#define  OMORIG_UPDATE_OBJ_ID_TAG     0x5013
#define  OMORIG_UPDATE_GRP_TAG        0x5014
#define  OMORI_FRWD_IOCTL_TAG         0x5015
#define  OMORI_ID_REQ_TAG             0x5017
#define  OMORI_CLASS_OID_REQ_TAG      0x5019
#define  OMORIG_UPDATE_GRP_DONE_TAG 0x5026
#define  OMORIG_UPDATE_OBJ_DONE_TAG 0x5028
```

7.3.3. API

7.3.3.1. Create / Destroy API

FUNCTION	EFFECT	ARGUMENTS

`omori_obj_register`	Registers object to the OM Database. Returns SUCCESS or FAILURE	CLASS ID, Object ID, pointer to object
`omori_obj_deregister`	Deregisters object from the OM Database. Returns SUCCESS or FAILURE	Object ID
`omori_assign_and_register_obj_id`	Registers object to the OM database, but requests OM to assign object ID to the registered object. Returns SUCCESS or FAILURE	CLASS ID, Object ID, pointer to object
`omori_create_obj`	Creates object of specified class in the specified group. If object of this class in the group already exists and unique flag is TRUE, then existing object ID will be returned, otherwise new object will be created.	VPN ID, VR ID, GROUP ID, CLASS ID, address space id, unique flag. Returns object id.
`omori_create_active_obj`	Same as omori_create_obj but after successful creation sends ACTIVATE_OBJECT IOCTL to the object	VPN ID, VR ID, GROUP ID, CLASS ID, address space id, unique flag. Returns object id.
`omori_destroy_obj`	Sends STOP_OBJECT IOCTL to the object if it was in ACTIVE state and after this destroys object by sending DESTROY_OBJECT IOCTL to the object and then destroying all management information regarding this object.	Object ID

7.3.3.2. Group Membership Change

FUNCTION	EFFECT	ARGUMENTS
`omori_add_obj_to group`	Adds object to the specified group. If object reviously belong to another group it removes from the first group and add to the new. Returns SUCCESS or FAILURE	Object ID, GROUP ID
`omorig_remove_obj_id`	Removes object from the group where it belongs and moves it to the group 0 (OM_BASE_GROUP). Returns SUCCESS or FAILURE	Object ID

7.3.3.3. Attribute Access

FUNCTION	EFFECT	ARGUMENTS

`omori_get_obj_group`	Returns Group ID of the group to which object belongs, if object is not local then information is retrieved from the OMORIG	Object ID
`omori_get_address_space`	Returns address space ID of the group to which object belongs, if object is not local then information is retrieved from the OMORIG	Object ID
`omori_get_local_obj_reference`	Returns pointer to the local object and NULL if object is not local	Object ID

7.4. Transaction layer

7.4.1. Data Structures

```
typedef
struct om_request_s
{
   TAG_DECL;                                  /* run time type identification */
   dlcl_list_t       list;                    /* List of requests */
   int               index;                   /* matching index */
   om_callback_f     *func;                   /* callback function */
   void              *cookie;
   int               tag;
   address_space_t   dest;                    /* Destination address */
}  om_request_t;
typedef
struct om_req_cookie_s
{
   TAG_DECL;                               /* run time type identification */
   int                  req_id;            /* Request ID */
   address_space_t      addr;              /* Address of the request originator */
   int                  tag;               /* message tag */
}  om_req_cookie_t;
```

7.4.2. API

FUNCTION	EFFECT	ARGUMENTS
`om_send_request`	Sends data to the destination, adds transaction control header in front of the packet, allocates a request, adds it to the request list. If callback function is NULL then free request.	Data packet, destination address, tag, callback function, request cookie.
`om_recv_request`	Receive request, allocate request cookie, save request id and source	Data packet

	address in it.	
`om_send_reply`	Sets message tag, depending on the channel, which data sent on. Sends data to the destination, adds transaction control header in front of the packet, free request cookie	Data packet, request cookie, tag, channels to send reply on.
`om_recv_reply`	Receive reply, calls callback function from the transaction control block (om_request_t). After completion free the request.	Data packet
`om_get_request_tag`	Retrieves message tag of the request.	Request cookie
`om_get_request_src`	Retrieves address of the originator of the request.	Request cookie
`om_abort_requests_by_addr`	Abort all the transactions associated with the address.	Address of the peer